Spring Boot Hackathon: Vehicle and Dealer Management System (90 Minutes)

Objective:

Build a RESTful API application using Spring Boot that allows CRUD operations and additional functionalities on Vehicles and Dealers for TVS Motor Company. The application should perform the following key operations using a MySQL or MongoDB database.

Entities:

1. Vehicle Entity:

- vehicleId (Primary Key)
- model (String)
- type (String) – e.g., Scooter, Bike
- price (Decimal)
- dealerId (Foreign Key) – reference to Dealer
- status (String) – e.g., Available, Sold
- manufactureDate (Date)

2. Dealer Entity:

- dealerId (Primary Key)
- name (String)
- location (String)
- contactNumber (String)
- email (String)
- registeredDate (Date)

Functionalities (Service Layer):

1. Add New Vehicle:

- Functionality to add a new vehicle to the system by assigning it to a dealer. The vehicle will have attributes like model, type, price, and manufacture date.
- Method: addVehicle(Vehicle vehicle)
- API: POST /api/vehicles

2. Add New Dealer:

- Functionality to register a new dealer with attributes like name, location, contact number, and email.

- Method: addDealer(Dealer dealer)
- API: POST /api/dealers

3. Get All Vehicles by Dealer:
    - Retrieve all vehicles registered with a particular dealer using the dealer's ID.
    - Method: getVehiclesByDealerId(Long dealerId)
    - API: GET /api/dealers/{dealerId}/vehicles
4. Update Vehicle Price:
    - Functionality to update the price of a particular vehicle using its ID.
    - Method: updateVehiclePrice(Long vehicleId, BigDecimal price)
    - API: PUT /api/vehicles/{vehicleId}/price
5. Mark Vehicle as Sold:
    - Mark a vehicle as sold by updating its status attribute.
    - Method: markVehicleAsSold(Long vehicleId)
    - API: PUT /api/vehicles/{vehicleId}/status/sold
6. Get All Available Vehicles:
    - Retrieve all vehicles that are available (not sold).
    - Method: getAvailableVehicles()
    - API: GET /api/vehicles/available
7. Delete Vehicle:
    - Functionality to delete a vehicle from the system using its ID.
    - Method: deleteVehicleById(Long vehicleId)
    - API: DELETE /api/vehicles/{vehicleId}
8. Search Dealer by Location:
    - Retrieve a list of dealers based on a specific location.
    - Method: searchDealerByLocation(String location)
    - API: GET /api/dealers/search/location/{location}
9. Get All Dealers:
    - Functionality to retrieve a list of all dealers in the system.
    - Method: getAllDealers()
    - API: GET /api/dealers
10. Update Dealer Contact Information:

- Update the contact number or email of a dealer by their ID.
- Method: updateDealerContact(Long dealerId, String contactNumber, String email)
- API: PUT /api/dealers/{dealerId}/contact

Database Considerations: (Any One)

- MySQL Database: You would need tables for vehicles and dealers with appropriate foreign key constraints between them.
- MongoDB: You can use separate collections for vehicles and dealers, with a reference to the dealer in the vehicle collection.

Example Use Case Flow:

1. Register a dealer.
2. Add a vehicle under that dealer.
3. Retrieve all vehicles for that dealer.
4. Update the price of a vehicle.
5. Mark a vehicle as sold.
6. Search for dealers in a specific location.

Note:- Create a Custom Exception IDNotFoundException , InValidVehicleException, Throw Where ever is needed.